



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра Инструментального и прикладного программного обеспечения
(ИиППО)

**ОТЧЁТ
ПО ПРАКТИЧЕСКИМ РАБОТАМ
по дисциплине
«Программирование на языке Java»**

Выполнил студент группы ИКБО-04-20

Хан А. А.

Принял ассистент кафедры ИиППО

Ермаков С.Р.

Практические работы выполнены «__»_____2021 г.

«Зачтено» «__»_____2021 г.

Москва
2021

Практическая работа №1. Основа

Язык Java - это объектно-ориентированный язык программирования. Программы, написанные на Java могут выполняться на различных операционных системах при наличии необходимого ПО - Java Runtime Environment.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task1>

Вывод программы

Opt №1

```
Sum(for):60  
Sum(while):60  
Sum(do while):60
```

Opt №3

```
1 1 1 1 1 1 1 1 1  
1 + - + - + - + - + - + - + - + --  
2 3 4 5 6 7 8 9 10
```

Opt №4

```
Массив из случайных чисел: -189 -789 -907 -129 -503 -977  
Отсортированный массив: -977 -907 -789 -503 -189 -129
```

Opt №5

```
C:\Users\Anastasia\.jdk\openjdk-16.0.2\bin\java.exe ...  
6227020800
```

Вывод

В результате выполнения работы, я получила практические навыки разработки программ, изучила синтаксис языка Java, освоила основные конструкций языка Java (циклы, условия, создание переменных и массивов, создание методов, вызов методов), а также научилась осуществлять стандартный ввод/вывод данных.

Практическая работа №2. Объектно-ориентированное программирование на Java

Язык Java - объектно-ориентированный язык программирования. В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Соккрытие данных называется инкапсуляцией.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task2>

Вывод программы

Ball

```
20 this is the size of the ball
100 this is the size of the ball
the colour and size of the ball is pink and 20
the colour and size of the ball is white and 100
```

Book

```
Hopkins, the author of the book with the number of pages 369
Smith, the author of the book with the number of pages 1
Hopkins the author of the book with the number of pages 369
Smith the author of the book with the number of pages 1
```

Dog

```
Bobo, age 13, species Maltese, owner Anna
Fafa, age 5, species Labrador, owner John
Coco, age 1, species Beagle, owner Monica
Bobo's age in human years is 91 years
Fafa's age in human years is 35 years
Coco's age in human years is 7 years
```

Вывод

В результате выполнения работы, я получила практические навыков разработки программ, изучила синтаксис языка Java, освоила основные конструкций языка Java (циклы, условия, создание переменных и массивов,

создание методов, вызов методов), а также научилась осуществлять стандартный ввод/вывод данных.

Практическая работа №3. Классы

В Java, класс является определением объектов одного и того же вида. Другими словами, класс — это тип данных, создаваемый программистом для решения задач. Он представляет из себя шаблон, или прототип, который определяет и описывает статические свойства и динамическое поведение, общие для всех объектов одного и того же вида.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task3>

Вывод программы

Book

```
Author: = Anna Smith  
Title = Romeo  
Year = 2000  
  
Author: = Elizabeth Pica  
Title = Juliet  
Year = 2003  
  
Author: = Alexandra Jojo  
Title = Me before you  
Year = 2010  
  
Author: = Mikhail Ham  
Title = The king  
Year = 1989  
  
Author: = Vladimir Gogo  
Title = The forever war  
Year = 1990
```

Human

```
Anna has 2 hands, 1 head and 2 legs  
Mark has 2 hands, 1 head and 2 legs  
Sophia has 2 hands, 1 head and 2 legs  
Anna has 2 hands, 1 head and 2 legs  
Mark has 2 hands, 1 head and 2 legs  
Sophia has 2 hands, 1 head and 2 legs
```

Вывод

В результате выполнения работы, я изучила основные концепции объектно-ориентированного программирования, понятие класса и научилась создавать классы.

Практическая работа №4. UML

Язык моделирования Unified Modeling Language (UML) является стандартом де-факто с 1998 года для проектирования и документирования объектно-ориентированных программ. Средствами UML в виде диаграмм можно графически изобразить класс и экземпляр класса.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task4>

Вывод программы

Author

```
Author:
Name = Anna
Email = annushka95@gmail.com
Gender = F
Author:
Name = Elizabeth
Email = zizi2005@mail.ru
Gender = F
Author:
Name = Alexandra
Email = sashulya2506@gmail.ru
Gender = F
Author:
Name = Mikhail
Email = misha0605@mail.com
Gender = M
Author:
Name = Vladimir
Email = vova.designer@gmail.ru
Gender = M
```

Ball

```
Ball @ (500.0, 90.0).
Ball @ (30.0, 70.0).
Ball @ (125.0, 15.0).
Ball @ (513.0, 99.0).
Ball @ (85.0, 143.0).
Ball @ (153.0, 51.0).
```

Вывод

В результате выполнения работы, я получила практические навыки работы с UML-диаграммами классов.

Практическая работа №5. Наследование, абстрактные классы

Одним из ключевых аспектов объектно-ориентированного программирования является наследование. С помощью наследования можно расширить функционал уже имеющихся классов за счет добавления нового функционала или изменения старого.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task5>

Вывод программы

Dogs

```
Имя: Jason  
Порода: Bobtail  
  
Имя: Nicky  
Порода: Bulldog  
  
Имя: Zizi  
Порода: Pug
```

Furniture

```
Магазин: Furniture  
Название: Divanchick  
Локация: Moscow  
  
Вид: Gaming chair  
Материал: Leather  
  
Вид: Corner sofa  
Материал: Fabric  
  
Вид: Kitchen table  
Материал: Wooden
```

Вывод

В результате выполнения работы, я изучила понятие наследования, и научилась реализовывать наследование в Java.

Практическая работа №6. Интерфейсы

Механизм наследования очень удобен, но он имеет свои ограничения. В частности, мы можем наследовать только от одного класса.

В языке Java подобную проблему позволяют решить интерфейсы. Интерфейсы определяют некоторый функционал, не имеющий конкретной реализации, который затем реализуют классы, применяющие эти интерфейсы. И один класс может применить множество интерфейсов.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task6>

Вывод программы

Opt №1

```
Сущность Voice assistant имеет имя ALISA  
  
Сущность Planet имеет имя MARS  
  
Сущность имеет имя 'ALISA'  
  
Сущность имеет имя 'MARS'
```

Opt №2

```
Аппарат фирмы 'Samsung', модель Galaxy A80, стоит в этом году 60000 рублей  
  
Аппарат фирмы 'Apple', модель Iphone 13, стоит в этом году 160000 рублей  
  
Сущность стоит 60000 рублей  
Сущность стоит 160000 рублей
```

Вывод

В результате выполнения работы, я на практике изучила понятие интерфейса, научилась создавать интерфейсы в Java и применять их в программах.

Практическая работа №7. Абстрактный суперкласс, подклассы

Класс, содержащий абстрактные методы, называется абстрактным классом. Такие классы при определении помечаются ключевым словом `abstract`.

Абстрактный метод внутри абстрактного класса не имеет тела, только прототип.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task7>

Вывод программы

Opt №2

```
Circle:
S=25446,900494
P=565,486678
Color=Pink
Filled = false
```

```
Rectangle:
S=900,000000
P=120,000000
Color=Orange
Filled = true
```

```
Circle:
S=25446,900494
P=565,486678
Color=Pink
Filled = false
```

```
Rectangle:
S=900,000000
P=120,000000
Color=Orange
Filled = false
```

```
Circle:
S=25446,900494
P=565,486678
Color=Pink
Filled = true
```

```
Rectangle:
S=900,000000
P=120,000000
Color=Orange
Filled = false
```

```
Circle:
S=25446,900494
P=565,486678
Color=Pink
Filled = false
```

```
Rectangle:
S=900,000000
P=120,000000
Color=Orange
Filled = true
```

Вывод

В результате выполнения работы, я на практике освоила работу с абстрактными классами и наследованием на Java.

Практическая работа №8. GUI

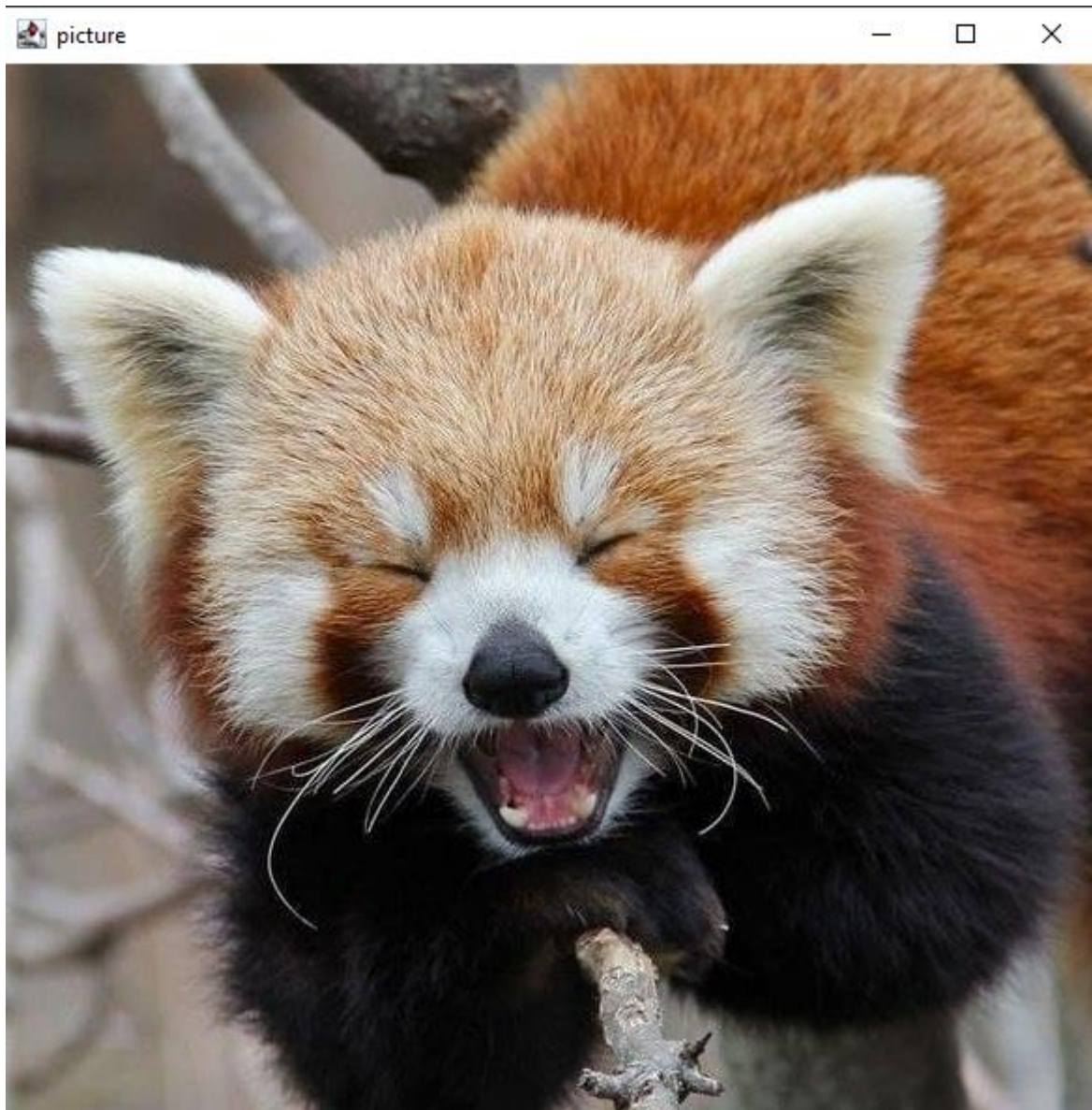
Для создания графического интерфейса пользователя можно использовать стандартную Java библиотеку Swing или AWT. В этих библиотеках имеются различные классы, позволяющие создавать окна, кнопки, текстовые поля, меню и другие объекты.

Программный код

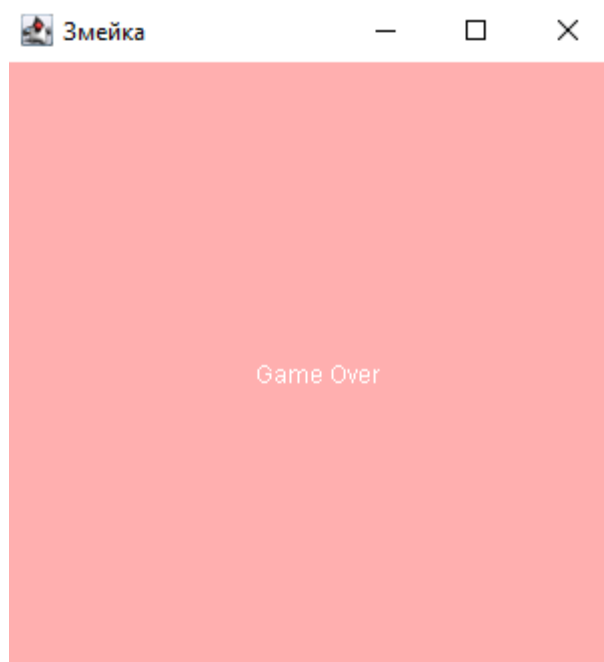
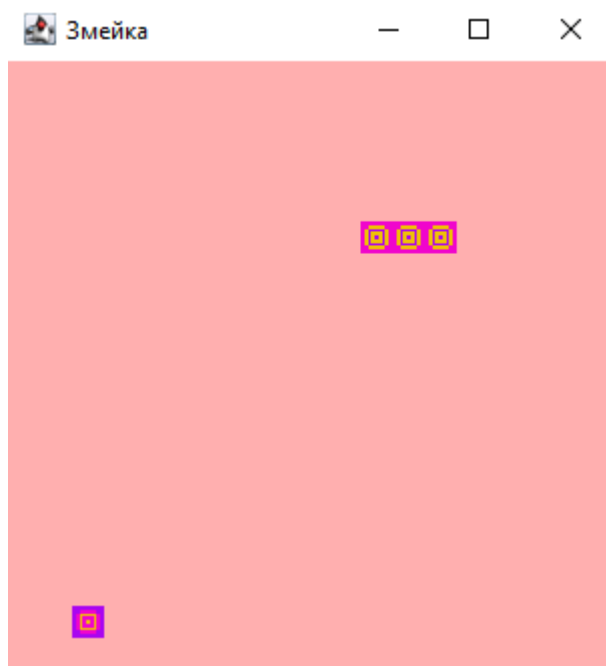
<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task8>

Вывод программы

Opt №2



Opt №3



Вывод

В результате выполнения работы, я на практике научилась создавать графический интерфейс пользователя, освоила работу с различными объектами для создания ГИП, менеджерами размещения компонентов.

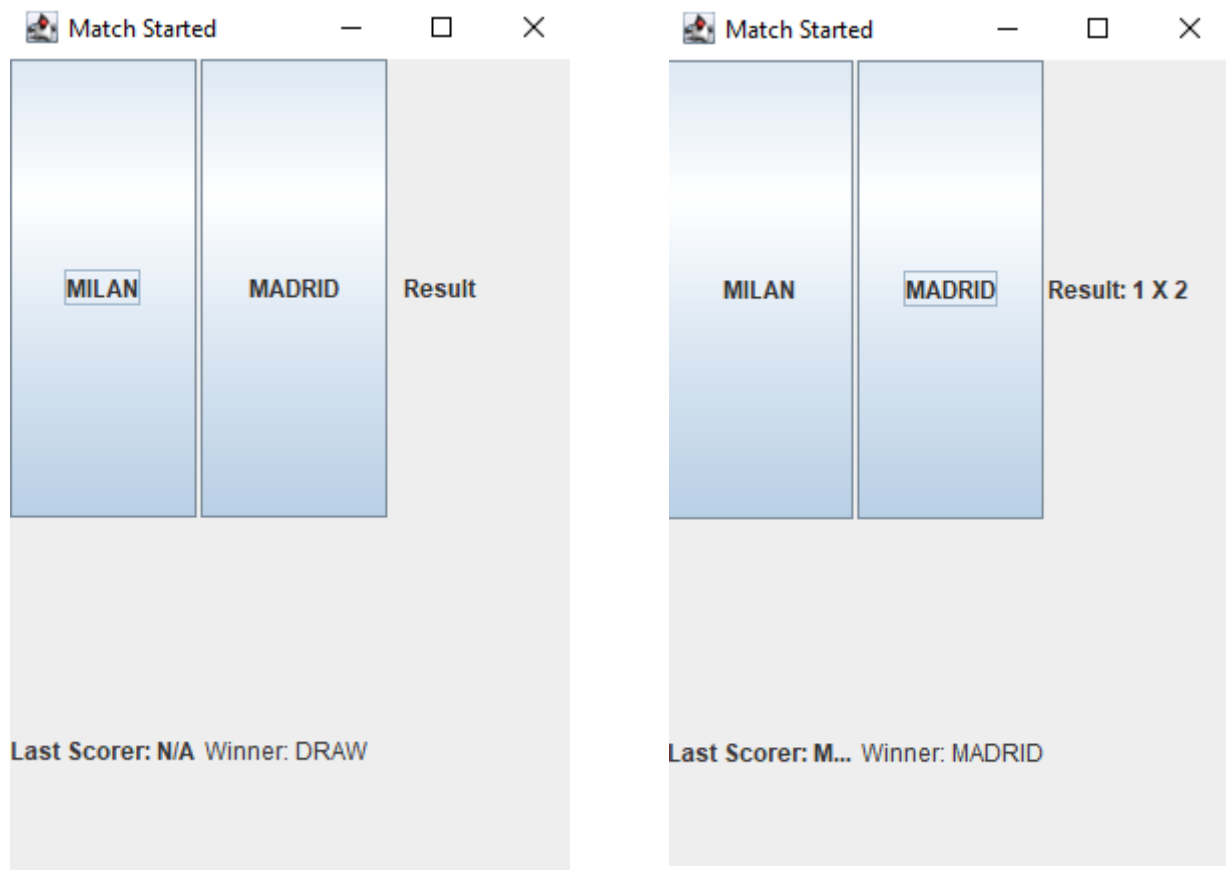
Практическая работа №9. Событийное программирование

Событие — это изменение состояния. События передаются между собой и представляют этапы в каком-то процессе.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task9>

Вывод программы



Вывод

В результате выполнения работы, я закрепила практические навыки по созданию приложений на Java с использованием элементов GUI.

Практическая работа №10. Рекурсия

В контексте языка программирования рекурсия — это некий активный метод (или подпрограмма) вызываемый сам по себе непосредственно, или вызываемой другим методом (или подпрограммой) косвенно. В первую очередь надо понимать, что рекурсия — это своего рода перебор. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task10>

Вывод программы

Opt №11

```
C:\Users\Anastasia\.jdk\openjdk-16.0.2\bin\java.exe ...  
1 2 3 4 5 6 0 0  
6
```

Opt №12

```
C:\Users\Anastasia\.jdk\openjdk-16.0.2\bin\java.exe ...  
1  
2  
9  
3  
4  
5  
0  
5  
3  
9  
1
```

Opt №14

```
C:\Users\Anastasia\.jdk\openjdk-16.0.2\bin\java.exe ...  
1 2 3 4 5 6 7 8 9
```

Opt №15

```
C:\Users\Anastasia\.jdk\openjdk-16.0.2\bin\java.exe ...  
9 8 7 6 5 4 3 2 1
```


Вывод

В результате выполнения работы, я на практике научилась разработке и программированию рекурсивных алгоритмов на языке Java.

Практическая работа №11. Обработка событий (Swing)

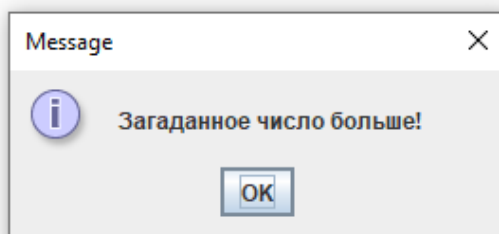
В контексте графического интерфейса пользователя наблюдаемыми объектами являются элементы управления: кнопки, флажки, меню и т.д. Они могут сообщить своим наблюдателям об определенных событиях, как элементарных (наведение мышкой, нажатие клавиши на клавиатуре), так и о высокоуровневых (изменение текста в текстовом поле, выбор нового элемента в выпадающем списке и т.д.).

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task11>

Вывод программы

Opt №1

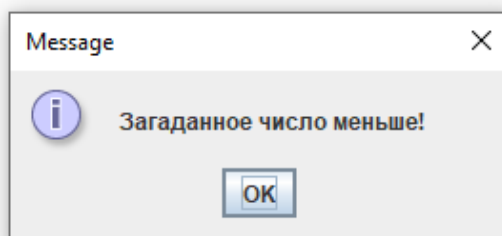


 Игра-угадайка



Угадайте число от 0 до 20

Использовать попытку

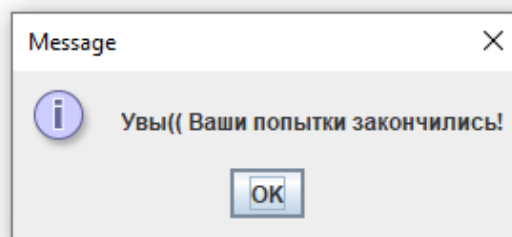


 Игра-угадайка




Угадайте число от 0 до 20

Использовать попытку



Opt №2



—

□

×

Черный

▼


Черный

Красный

Синий

Times New Roman

▼



—

□

×

Черный

▼

Times New Roman

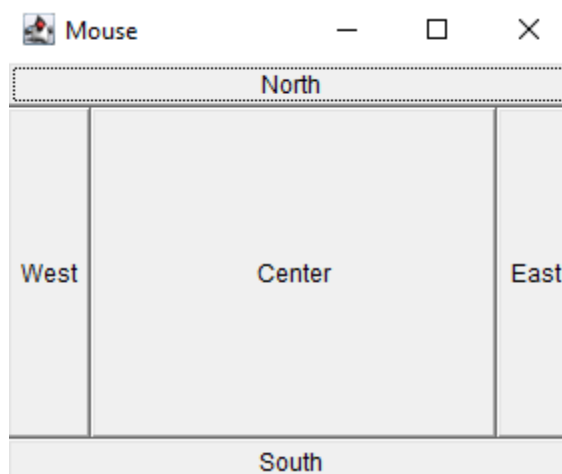
▼

Times New Roman

MS Sans Serif

Courier New

Opt №3



Вывод

В результате выполнения работы, я научилась обрабатывать различные события для разных компонентов (кнопок, меню и т. д.).

Практическая работа №12. Сортировка

Сортировка — это процесс упорядочивания списка элементов (организация в определенном порядке) исходного списка элементов, который возможно организован в виде контейнера или храниться в виде коллекции.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task12>

Вывод программы

```
Стандартная сортировка по id:  
[[id=5, name=Lena, age=20, GPA=75], [id=8, name=Alex, age=22, GPA=85], [id=9, name=Sasha, age=21, GPA=90], [id=10, name=Marina, age=19, GPA=80]]  
Comparator по GPA:  
[[id=9, name=Sasha, age=21, GPA=90], [id=8, name=Alex, age=22, GPA=85], [id=10, name=Marina, age=19, GPA=80], [id=5, name=Lena, age=20, GPA=75]]  
Comparator по возрасту:  
[[id=10, name=Marina, age=19, GPA=80], [id=5, name=Lena, age=20, GPA=75], [id=9, name=Sasha, age=21, GPA=90], [id=8, name=Alex, age=22, GPA=85]]  
Comparator по имени:  
[[id=8, name=Alex, age=22, GPA=85], [id=5, name=Lena, age=20, GPA=75], [id=10, name=Marina, age=19, GPA=80], [id=9, name=Sasha, age=21, GPA=90]]
```

Вывод

В результате выполнения работы, я на практике освоила методы сортировки с использованием приемов программирования на объектно-ориентированном языке Java.

Практическая работа №13. Коллекции, очереди, списки в Java

Для хранения наборов данных в Java предназначены массивы. Однако их не всегда удобно использовать, прежде всего потому, что они имеют фиксированную длину. Эту проблему в Java решают коллекции. Однако суть не только в гибких по размеру наборах объектов, но и в том, что классы коллекций реализуют различные алгоритмы и структуры данных.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task13>

Вывод программы

Opt №1

```
ArrayList{l=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, null, null, null, null, null, null, null, null, null, null, null, null, null, null], count=20, size=30}
ArrayList{l=[0, 1, 2, 3, 4, 6, 8, 10, 12, 14, 16, 18, null, null, null, null], count=12, size=16}
6

ArrayList{l=[null, null, null, null, null, null, null, null], count=0, size=8}
ArrayList{l=[String number 1, String number 2, String number 3, String number 4, String number 5, String number 6, String number 7, String number 8, String number 9, String number 10], count=10, size=10}

ArrayList{l=[String number 1, String number 2, String number 3, String number 4, String number 5, String number 6, String number 7, String number 8, String number 9, String number 10], count=10, size=10}
String number 7
String number 6
ArrayList{l=[String number 1, String number 2, String number 3, String number 4, String number 5, String number 7, String number 8, String number 9, String number 10], count=9, size=10}
ArrayList{l=[String number 1, String number 10, String number 2, String number 3, String number 4, String number 5, String number 5, String number 6, String number 7, String number 8, String number 9], count=11, size=11}
```

Opt №2

```
Enter the string
потоп
потоп is palindrome
```

Opt №3

```
Текущий список стран:  
Russia  
Germany  
Italy  
Ukraine
```

```
Новый список стран:  
Russia  
Germany  
Moldova  
Ukraine  
Poland  
В списке есть Германия!
```

```
Список студентов:  
Vanya  
Nikita  
Oleg
```

```
8-ой Элемент списка: 48  
Весь список:  
40 41 42 43 44 45 46 47 48 49  
Удален 5-ый элемент списка.  
Новый список:  
40 41 42 43 44 46 47 48 49
```

Вывод

В результате выполнения работы, я на практике изучила различные коллекции в Java.

Практическая работа №14. Контейнерные классы

Java Collections Framework — это набор связанных классов и интерфейсов, реализующих широко используемые структуры данных — коллекции. На вершине иерархии в Java Collection Framework располагаются 2 интерфейса: Collection и Map. Эти интерфейсы разделяют все коллекции, входящие в фреймворк на две части по типу хранения данных: простые последовательные наборы элементов и наборы пар «ключ — значение» (словари).

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task14>

Вывод программы

Opt №1

```
Карты не могут повторяться в колодах!  
Введите карты первой колоды(0-9):  
1 5 6 7 9  
Введите карты второй колоды(0-9):  
2 3 4 8 0  
Первая колода: [1, 5, 6, 7, 9]  
Вторая колода: [2, 3, 4, 8, 0]  
Ботва! Количество ходов: 106
```

```
Карты не могут повторяться в колодах!  
Введите карты первой колоды(0-9):  
0 5 9 7 3  
Введите карты второй колоды(0-9):  
1 2 4 6 8  
Первая колода: [0, 5, 9, 7, 3]  
Вторая колода: [1, 2, 4, 6, 8]  
Победа второго игрока! Количество ходов: 43
```

Opt №2

```
Правила игры в 'пьяницу':
1) В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта побеждает меньшую, карта «0» побеждает карту «9».
2) В этой игре карточная колода раздается поровну двум игрокам.
3) Далее они открывают по одной верхней карте, и тот, чья карта старше, забирает себе обе открытые карты, которые кладутся под низ его колоды. Тот,
4) Для простоты будем считать, что все карты различны по номиналу и что самая младшая карта побеждает самую старшую карту ("шестерка берет туза").
5) Игрок, который забирает себе карты, сначала кладет под низ своей колоды карту первого игрока, затем карту второго игрока (то есть карта второго
С использованием стека
Введите карты первого игрока
1 2 5 9 0
Введите карты второго игрока
8 3 4 7 6
Результат: first 25
```

Opt №3

```
С использованием очереди
Введите карты первого игрока
1 2 5 9 0
Введите карты второго игрока
8 3 4 7 6
Результат: first 17
```

Opt №4

```
С использоованием двухсторонней очереди
Введите карты первого игрока
1 2 5 9 0
Введите карты второго игрока
8 3 4 7 6
Результат: first 17
```

Opt №5

```
С использованием двусвязного списка
Введите карты первого игрока
1 2 5 9 0
Введите карты второго игрока
8 3 4 7 6
Результат: first 17
```

Вывод

В результате выполнения работы, я на практике изучила приемы работы со стандартными контейнерными классами Java Collection Framework.

Практическая работа №15. Работа с файлами

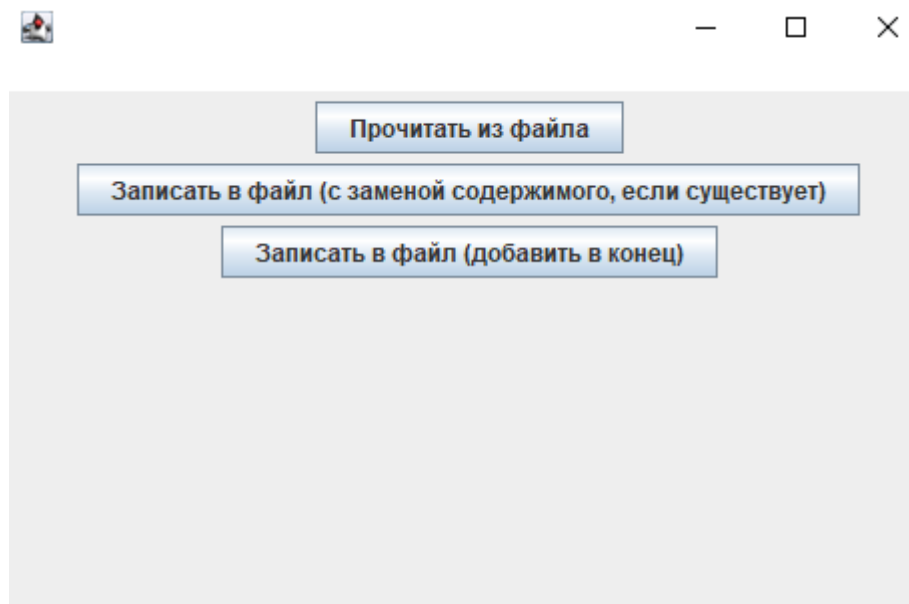
Класс `FileWriter` является производным от класса `Writer`. Он используется для записи текстовых файлов.

Класс `FileReader` наследуется от абстрактного класса `Reader` и предоставляет функциональность для чтения текстовых файлов.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task15>

Вывод программы



Вывод

В результате выполнения работы, я на практике освоить работу с файлами на языке Java, получила практические навыки по чтению и записи данных в файл.

Практическая работа №16. Индивидуальный проект

Выбранный мною проект, связан с медициной. Данное приложение будет направлено на помощь людям, которые собираются на первичный прием, но не знают, что для этого нужно сделать.

Помощь будет заключаться в следующем:

1. Список необходимых обследований для первичного приема, по категориям врачей;
2. Места, в которых пациент может пройти обследование.

Что будет лежать в основе:

1. Обследования:

Основные:

- 1.Анализы
- 2.Флюорография
- 3.ЭКГ

Дополнительные:

- 1.УЗИ
- 2.КТ
- 3.МРТ

2. Различные виды услуг, которые будут зависеть от категории обследования.

3. Места, где можно сделать данную услугу.

Все виды обследований будут подразделяться на различные категории. Для удобства использования, категории будут разделены на возрастные категории и врачей.

Работа пользователя с приложением будет заключаться в следующем:

1. Пользователь заходит в приложение, выбирает возрастную категорию, которой он соответствует;
2. Следом будет выбор специалиста;

3. После выбора основных категорий, пользователю будет показан перечень обследований, основных и дополнительных, на основе которых он может подготовиться к приему;

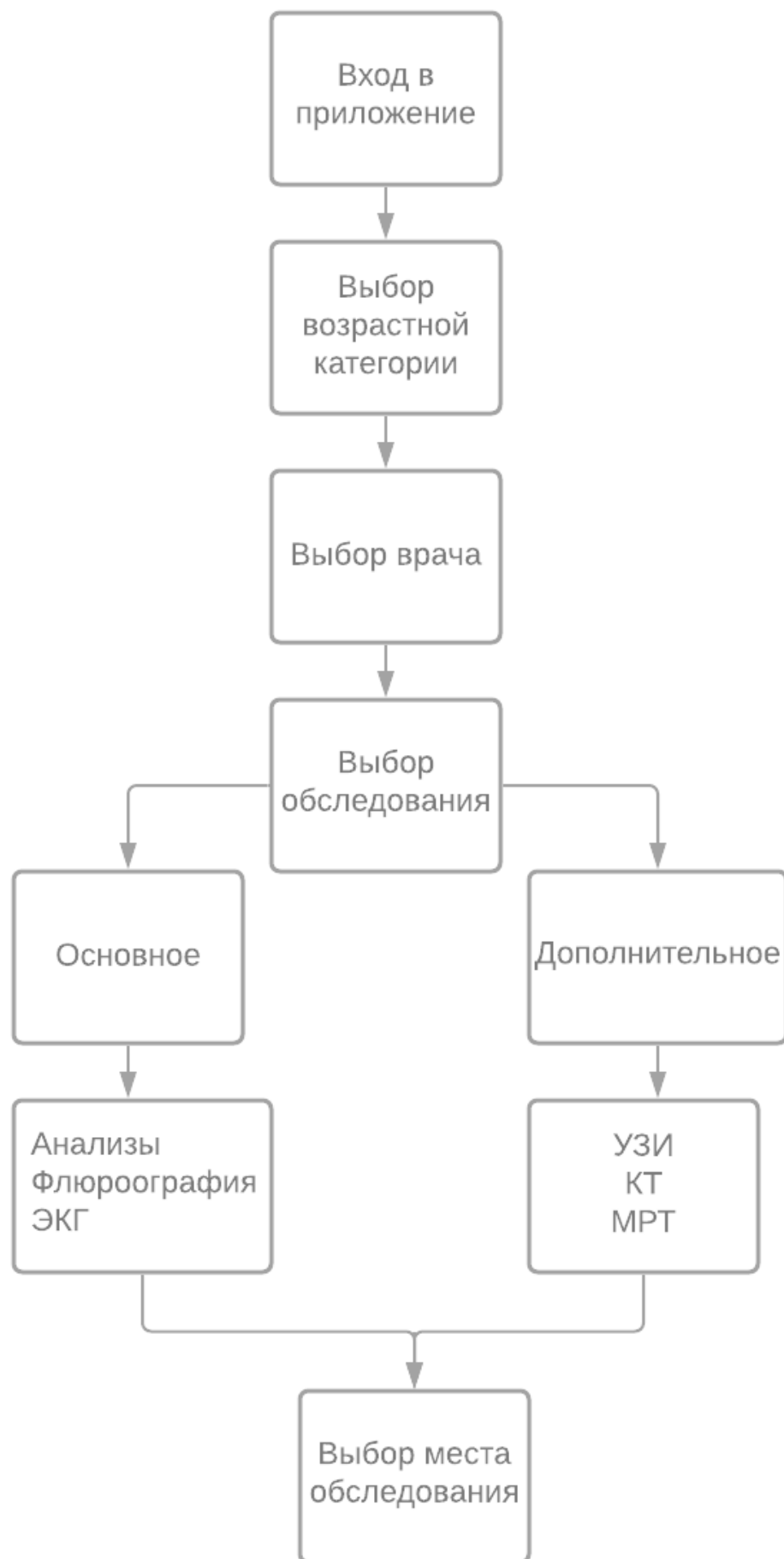
4. Последним этапом будет выбор места, в котором пользователь сможет пройти обследование на выгодных для него условиях;

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task16>

Вывод программы

**Схема работы
приложения:**



Пример приложения:



Вывод

В результате выполнения работы, я составила примерный план реализации приложения, которого хочу создать.

Практическая работа №17. Перечисления

Программируя мы часто сталкиваемся с необходимостью ограничить множество допустимых значений для некоторого типа данных. Для решения подобных задач во многих языках программирования со статической типизацией предусмотрен специальный тип данных – перечисление enum.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task17>

Вывод программы

Basa

```
Переменная bas содержит Examination.  
  
Обследования включают в себя 2 категории:  
  
1.Основные обследования:  
  
1.Анализы:  
  
-Общий анализ крови;  
-Общий анализ мочи;  
-Биохимия развернутая;  
-Коагулограмма скрининг;  
-С-реактивный белок;  
-ТТГ,Т3,Т4  
  
2.Флюорография;  
  
3.ЭКГ;  
  
2.Дополнительные обследования:
```

2.Дополнительные обследования:

1.УЗИ:

- УЗИ предстательной железы;
- УЗИ в педиатрии;
- УЗИ сосудов;
- УЗИ мочевыделительной системы;
- УЗИ щитовидной железы;
- УЗИ органов брюшной полости;
- УЗИ органов малого таза;
- УЗИ при беременности

2.КТ:

- КТ головы;
- КТ с контрастом;
- КТ позвоночника;
- КТ суставов и костей;
- КТ внутренних органов;
- КТ мягких тканей

2.КТ:

- КТ головы;
- КТ с контрастом;
- КТ позвоночника;
- КТ суставов и костей;
- КТ внутренних органов;
- КТ мягких тканей

3.МРТ:

- МРТ внутренних органов;
- МРТ головы;
- МРТ мягких тканей;
- МРТ периферической нервной системы;
- МРТ при беременности;
- МРТ позвоночника;
- МРТ суставов;
- МРТ с контрастом

Basa2

Константы перечислимого типа Basa:

Age

Doctors

Examination

Place

Константы перечислимого типа Basa:

Age Doctors Examination Place

Переменная bas содержит Examination

Basa3

```
Получить все константы и порядковые значения:  
Age 0  
Doctors 1  
Examination 2  
Place 3  
  
Продemonстрировать применение метода compareTo():  
Age предшествует Doctors  
  
Продemonстрировать применение метода equals():  
  
Сравнить на равенство:
```

Answer

```
Место  
Врач  
Место  
Место  
Место
```

Examination

Цена на процедуру:

УЗИ внутренних органов стоит 1800 рублей

Цены на обследования:

GeneralBloodTest стоит 300 рублей

GeneralUrineAnalysis стоит 250 рублей

BiochemistryExpanded стоит 1600 рублей

CoagulogramScreening стоит 950 рублей

CRActiveProtein стоит 350 рублей

TSH стоит 400 рублей

T3 стоит 450 рублей

T4 стоит 450 рублей

Fluorography стоит 1000 рублей

ECG стоит 1000 рублей

UltrasoundOfThePelvicOrgans стоит 1900 рублей

UltrasoundOfTheBristleGland стоит 2500 рублей

UltrasoundOfBloodVessels стоит 2000 рублей

UltrasoundOfTheAbdominalCavity стоит 1800 рублей

UltrasoundDuringPregnancy стоит 1900 рублей

UltrasoundOfTheUrinarySystem стоит 2200 рублей

CTOfTheHead стоит 4200 рублей

CTWithContrast стоит 3900 рублей

CTOfTheSpine стоит 9800 рублей

CTWithContrast стоит 3900 рублей

CTOfTheSpine стоит 9800 рублей

CTOfJointsAndBones стоит 3700 рублей

CTOfInternalOrgans стоит 4200 рублей

CTOfSoftTissues стоит 3200 рублей

MRIOfTheHead стоит 5000 рублей

MRIOfInternalOrgans стоит 8000 рублей

MRIOfSoftTissues стоит 5000 рублей

MRIOfThePeripheralNervousSystem стоит 5000 рублей

MRIDuringPregnancy стоит 14000 рублей

MRIOfTheSpine стоит 14000 рублей

MRIIOfJoints стоит 6000 рублей

MRIWithContrast стоит 5000 рублей

Вывод

В результате выполнения работы, я научилась с большим количеством данных, которые перечисляются, выводятся, и передают константы.

Практическая работа №18. Обработка исключений

В мире программирования возникновение ошибок и непредвиденных ситуаций при выполнении программы называют исключением. В программе исключения могут возникать в результате неправильных действий пользователя, отсутствии необходимого ресурса на диске, или потери соединения с сервером по сети.

Причинами исключений при выполнении программы также могут быть ошибки программирования или неправильное использование API. В отличие от нашего мира, программа должна четко знать, как поступать в такой ситуации. Для этого в Java предусмотрен механизм исключений.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task18>

Вывод программы

Location

```
Введите первую локацию 1 :  
KDL  
Введите вторую локацию 2 :  
INVITRO  
Изменение в локации  
Вероятность одинаковой цены на услуги в обеих локациях  
Новая локация - локация 1: INVITRO локация 2: INVITRO
```

Name

```
Введите имя :  
Alexa  
Введите возраст :  
25  
Пациент :  
Имя : Alexa  
Возраст : 25
```

Zero

```
Введите числитель :  
25  
Введите знаменатель :  
86  
Результат = 0.29069766
```

Вывод

В результате выполнения работы, я научилась работать с исключениями, обрабатывать различные ситуации, которые могут произойти.

Практическая работа №19. Создание пользовательских исключений

Исключения - это те же классы и объекты. И иногда удобно выстроить свою иерархию исключений, заточенных под конкретную задачу. Дабы более гибко обрабатывать и реагировать на те исключительные ситуации, которые специфичны решаемой задаче.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task19>

Вывод программы

```
Найдено исключение  
Программа работает  
123456789
```

Вывод

В результате выполнения работы, я научилась создавать, работать с пользовательскими исключениями, которые пользователь сможет подстроить под конкретную задачу.

Практическая работа №20. Работа с датой и временем

Обработка дат и времени довольно сложная задача. В мире существует большое количество часовых поясов, которые периодически меняются. Нужно учитывать переход на зимнее и летнее время, секунды координации, високосные года и многое другое.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task20>

Вывод программы

```
2021-12-14T19:47:11.277007600  
  
Tue, 14 Dec 2021 19:47:11 +0300  
  
1639500431288  
Tue Dec 14 19:47:11 MSK 2021  
  
java.util.GregorianCalendar[time=1639500431314,areF  
  
14. 12. 2021  
  
2021/12/14 19:47:11
```

Вывод

В результате выполнения работы, я научилась работать с датой и временем, а именно, с временными зонами, такими как, текущая дата и время на момент создания объекта.

Практическая работа №21. Дженерики

Дженерики (обобщения) — это особые средства языка Java для реализации обобщённого программирования: особого подхода к описанию данных и алгоритмов, позволяющего работать с различными типами данных без изменения их описания.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task21>

Вывод программы

```
class ru.mirea.task21.Objects.ObjectHolder
ru.mirea.task21.Objects.Doctor@16b98e56
[ru.mirea.task21.Objects.CustomObject@7ef20235]
Doctor object

=====

class ru.mirea.task21.Objects.ObjectHolder
ru.mirea.task21.Objects.Checkup@4f3f5b24
[ru.mirea.task21.Objects.CustomObject@15aeb7ab]
Checkup object
```

Вывод

В результате выполнения работы, я получила практические навыки работы с дженериками. Поняла, что дженерики – это сильная проверка типов во время компиляции и устранение необходимости явного приведения.

Практическая работа №22. Паттерн Factory

Фабрика — это шаблон проектирования, который помогает решить проблему создания различных объектов в зависимости от некоторых условий.

Шаблон проектирования Фабрика позволяет управлять созданием объектов.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task22>

Вывод программы

```
Детали по обследованию: КТ  
Количество обследований = 1 | Цена = 4200 рублей  
Детали по обследованию: УЗИ  
Количество обследований = 3 | Цена = 7000 рублей  
Детали по обследованию: МРТ  
Количество обследований = 2 | Цена = 20000 рублей
```

Вывод

В результате выполнения работы, я получила навыки работы с фабрикой, научилась изменять тип создаваемых объектов.

Практическая работа №23. Обработка строк в Java, hashCode()

Хэш - это некоторое число, генерируемое на основе объекта и описывающее его состояние.

Метод hashCode, наряду с equals, играет важную роль в сравнении объектов. По сути он показывает изменилось ли состояние объекта, которое мы используем в equals для сравнения.

Если планируется использовать объекты в качестве ключа в ассоциативном массиве, то переопределять hashCode обязательно. Если в классе переопределяется equals, то обязательно надо переопределять hashCode и наоборот.

Эти методы всегда должны определяться парой!

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task23>

Вывод программы

```
599280361
-1407390601
false
false
true
```

Вывод

В результате выполнения работы, я получила навыки работы с hashCode() и equals(), которые я использовала в своем индивидуальном проекте.

Практическая работа №24. Использование регулярных выражений для работы со строками

Регулярные выражения представляют собой похожий, но гораздо более сильный инструмент для поиска строк, проверки их на соответствие какому-либо шаблону и другой подобной работы. Англоязычное название этого инструмента — Regular Expressions или просто RegExp. Строго говоря, регулярные выражения — специальный язык для описания шаблонов строк.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task24>

Вывод программы

RegExp1

Данные о пациентах располагаются в файле queue.txt

```
queue: 5; Имя пациента: Мария; Возраст: 19; Диагноз: Двухсторонняя пневмония;
queue: 3; Имя пациента: Михаил; Возраст: 57; Диагноз: Инсульт;
queue: 1; Имя пациента: Виктория; Возраст: 28; Диагноз: Хронический пиелонефрит;
queue: 10; Имя пациента: Василий; Возраст: 35; Диагноз: Артроз;
queue: 9; Имя пациента: Ангелина; Возраст: 9; Диагноз: Острое респираторное заболевание;
```

```
Enter patient's place in the queue:
3
queue: 3; Имя пациента: Михаил; Возраст: 57; Диагноз: Инсульт;
```

RegExp2

```
def
defdefdefdfde
abc def w t g a w
    ho his prorm orks
bcdefhwthisrgrawrkssfddgdfg
13342465676
abc def ghi          abc def how this program works fdgdfg ascxf

abcdefghi12345566abcdefhowthisprogramworksfdgdfgasccxf2332
abcdefghi12345566abcdefhowthisprogramworksfdgdfgasccxf2332

BACDEFABCD E
\p{ASCII} All ASCII:[\x00-\x7F]
abc   def  ghi      ~                               1 2 3 4 55 66                jkl BACDEF mndn
[^\x00-\x7F] means non ASCII characters
± ± ± ¤ ¤ ¤ ¤ ¤ ¤ ¤ ¤ © © © © © © © ¢ ¢ ¢ ¢ ¢ ¢ ¢ ¢ † † † † † † † †
```

Вывод

В результате выполнения работы, я получила навыки работы с регулярными выражениями.

Практическая работа №25. Паттерн Decorator

Паттерн «Декоратор» позволяет динамически добавлять объекту новые обязанности, не прибегая при этом к порождению классов. При этом, работа с подобной структурой является более удобной и гибкой, нежели со множеством классов. Для этого, ссылка на декорируемый объект помещается в другой класс, называемый «Декоратором». Причем, и декоратор и декорируемый объект реализуют один и тот-же интерфейс, что позволяет вкладывать несколько декораторов друг в друга, добавляя тем самым декорируемому объекту любое число новых обязанностей. Декоратор переадресует внешние вызовы декорируемому объекту сопровождая их наложением дополнительных обязанностей.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task25>

Вывод программы

```
CT of the spine <> 9800
CT of the spine + with contrast + <> 13700
CT of the spine + with contrast + duplicate snapshot <> 14250
```

```
MRI of the head <> 9500
MRI of the head + with contrast + <> 13400
MRI of the head + with contrast + duplicate snapshot <> 13950
```

Вывод

В результате выполнения работы, я получила навыки работы с Паттерном «Декоратор».

Практическая работа №26. Паттерн Strategy

Стратегия — это поведенческий паттерн, выносит набор алгоритмов в собственные классы и делает их взаимозаменяемыми.

Другие объекты содержат ссылку на объект-стратегию и делегируют ей работу. Программа может подменить этот объект другим, если требуется иной способ решения задачи.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task26>

Вывод программы

Example

```
1750
1350
```

Strategy

```
Person02
Прохождение диагностики
CT of Head
MRI of Head
MRI of Head
CT of Spine
```

Вывод

В результате выполнения работы, я получила навыки работы с Паттерном «Стратегия».

Практическая работа №27. HashMap()

Мар - “ассоциативный массив”, или же более распространенные варианты “словарь”, “карта”.

Внутри Мар данные хранятся в формате “ключ”-”значение”, то есть по парам. И в качестве ключей, и в качестве значений могут выступать любые объекты — числа, строки или объекты других классов.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task27>

Вывод программы

```
Инициализированная карта:  
{1000=Диана, 1001=Алексей, 1002=Вадим, 1003=Анастасия, 1004=Галина, 1005=Вячеслав}  
Текущие размер карты: 6  
Карта после изменения:  
{1000=Диана, 1001=Алексей, 1002=Вадим, 1003=Анастасия, 1004=Галина, 1005=Вячеслав}  
Значение с ключом = 1004: null  
Карта после удаления ключа = 1001:  
{1000=Диана, 1002=Вадим, 1003=Анастасия, 1004=Галина, 1005=Вячеслав}
```

Вывод

В результате выполнения работы, я получила навыки работы с HashMap. Это очень универсальный прием, внутри массива хранятся данные, у каждого, уникальный ключ. И работать с ним очень удобно, когда большое количество данных.

Практическая работа №28. Анонимные и вложенные классы

Вложенные классы:

Класс называется вложенным (nested), если он определен внутри другого класса.

У таких классов есть три вида:

1) Вложенный нестатический класс

Нужен тогда, когда объект сложен, или комплексный, тогда его можно разделить на несколько подобъектов.

2) Статический вложенный класс

Нужен для использования извне. Используются вместе с public.

3) Вложенный класс в методе

Он имеет доступ и к полям(нестатическим), и к переменным метода(const).

Анонимные классы:

Анонимные классы – это вложенные без имени класса. Они обычно объявляются либо как подклассы существующего класса, либо как реализации некоторого интерфейса.

Определяются, когда они создаются.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task28>

Вывод программы

Вложенные классы

```
Checkup 1 is starting...
Patient 1 is starting...
```

Анонимные классы

```
Someone is treating...
```

Вывод

В результате выполнения работы, я получила навыки работы с вложенными и анонимными классами.

Практическая работа №29. Сериализация

Сериализация — это процесс сохранения состояния объекта в последовательность байт.

Любой Java-объект преобразуется в последовательность байт. В Java за процессы сериализации отвечает интерфейс `Serializable`. Этот интерфейс крайне прост: чтобы им пользоваться, не нужно реализовывать ни одного метода.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task29>

Вывод программы

```
ru.mirea.task29.Doctor[
  name = Alexandr Volokhov,
  price per operation = 80000.0,
  operationDay = Tue Dec 15 00:00:00 MSK 1987][
  sick = ru.mirea.task29.Patient[
    name = Denise Temiroff,
    price per operation = 55000.0,
    operationDay = Tue Oct 01 00:00:00 MSD 2002]]
ru.mirea.task29.Patient[
  name = Denise Temiroff,
  price per operation = 55000.0,
  operationDay = Tue Oct 01 00:00:00 MSD 2002]
```

Вывод

В результате выполнения работы, я получила навыки работы с сериализацией.

Практическая работа №30. JUnit

JUnit — это фреймворк автоматического тестирования кода, т. е. тестирования отдельных участков кода, например, методов или классов.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task30>

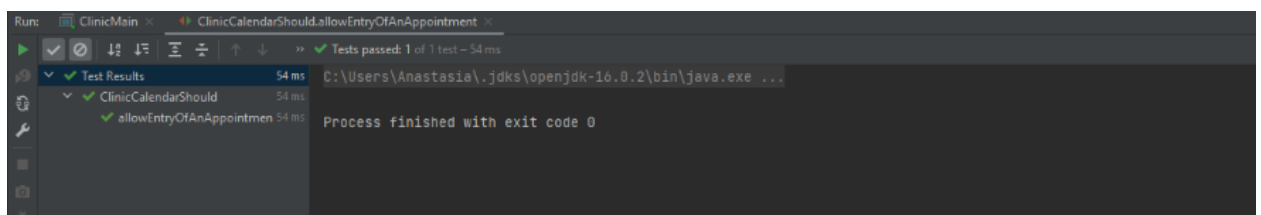
Вывод программы

```
Please select an option:
1. Enter a Patient Appointment
2. View All Appointments
X. Exit System.
Option: 1

Please Enter Appointment Info:
Patient Last Name: Naomi
Patient First Name: Smith
Appointment Date (M/d/yyyy h:m a): 1/3/2020 5:50 am
Doctor Last Name: Avery
Patient entered successfully.

Please select an option:
1. Enter a Patient Appointment
2. View All Appointments
X. Exit System.
Option: 2

All Appointments in System:
1/3/2020 05:50 AM: Naomi, Smith          Doctor: Jim Avery
```



Вывод

В результате выполнения работы, я получила навыки работы с JUnit. Опыт, полученный при работе с JUnit, важен в разработке концепций тестирования программного обеспечения.

Практическая работа №31. Тестирование

JUnit — это фреймворк автоматического тестирования кода, т. е. тестирования отдельных участков кода, например, методов или классов.

Программный код

<https://github.com/Nanochka1/JavaFirstMirea/tree/master/src/ru/mirea/task31>

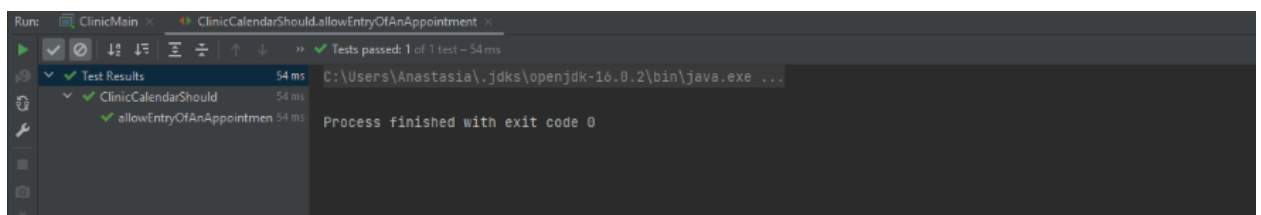
Вывод программы

```
Please select an option:
1. Enter a Patient Appointment
2. View All Appointments
X. Exit System.
Option: 1

Please Enter Appointment Info:
Patient Last Name: Naomi
Patient First Name: Smith
Appointment Date (M/d/yyyy h:m a): 1/3/2020 5:50 am
Doctor Last Name: Avery
Patient entered successfully.

Please select an option:
1. Enter a Patient Appointment
2. View All Appointments
X. Exit System.
Option: 2

All Appointments in System:
1/3/2020 05:50 AM: Naomi, Smith          Doctor: Jim Avery
```



Вывод

В результате выполнения работы, я получила навыки работы с JUnit. Опыт, полученный при работе с JUnit, важен в разработке концепций тестирования программного обеспечения.

Практическая работа №32. Защита

Приложение реализовано в программе Android Studio, с помощью ее внутренних библиотек.

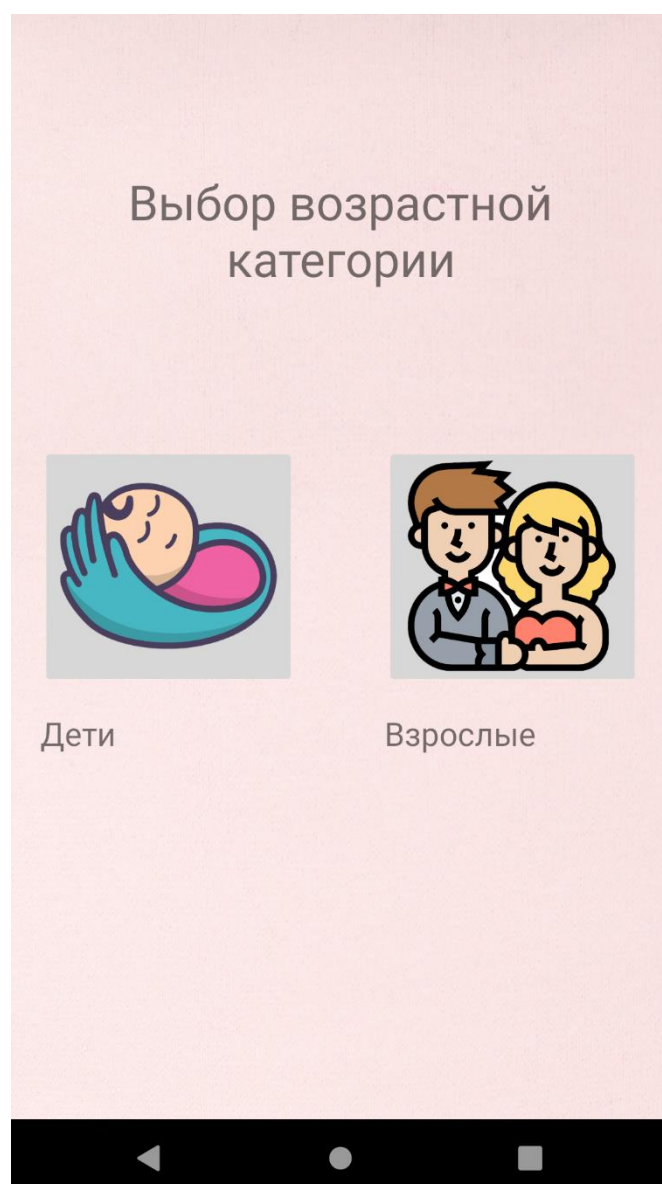
Темы, затронутые в приложении:

- Метод equals();
- Работа со временем;
- Перечисления;
- Графический интерфейс;
- Обработка событий при нажатии на кнопку;
- Методы Android;
- Коллекции, очереди и списки в Java;
- Наследование;
- Суперклассы.

Программный код

https://github.com/Nanochka1/Medicine_app

Вывод программы



Выбор специалиста

Терапевт
Кардиолог
Эндокринолог
олог
Хирург
Невролог
Гастроэнтеролог
Гинеколог
Уролог



Какие анализы нужны

Общий анализ крови
Общий анализ мочи
Биохимия развернутая
Коагулограмма скрининг
С - реактивный белок
ТТГ, Т4, Т3
Флюорография

Какие анализы нужны

Общий анализ крови
Общий анализ мочи
УЗИ органов брюшной
полости

г. Москва

Медицинские центры предоставляющие данные услуги



KDL

Часы работы 7:00 - 15:00
Телефон: +7 (800) 700-60-40



INVITRO

Часы работы 00:00 - 23:55
Телефон: +7 (495) 363-65-70



HELIX

Часы работы 7:00 - 19:00
Телефон: +7 (495) 374-65-70



Гемотест

Часы работы 7:00 - 19:30
Телефон: +7 (495) 476-50-97

Вывод

В результате выполнения проекта, я научилась работать с Android Studio. Создавать конфигурацию эмулятора, для трансляции проекта, работать с дизайном приложений. А также закрепила знания, полученные в ходе обучения, такие как, наследование, суперклассы, работа с кнопками и текстом и др.